Victorian 6502 User Group Newsletter

KAOS

For People Who Have Got Smart

OSI SYM KIM AIM APPLE UK101 ORANGE

Registered by Australia Post
Publication No. VBG4212

ADDRESS ALL CORRESPONDENCE TO 10 FORBES ST., ESSENDON, VIC. 3040

Vol.3 No.5 February 1983

It was decided at the last meeting to hold a 'Garage' sale at the February meeting. The sale will go from 1pm to 2pm or finish earlier if all sold out. Everyone bringing goods for sale will be responsible for them at all times, (ie. goods you bring and don't sell must be taken away).

There is no point in talking about renewing your membership at this time, if you haven't paid, then you're not reading this newsletter. (*Apologies to M.J.L. for pirating his material.*) On the same subject, there seems to be confusion amongst some members as to the yearly membership fee. As stated on the front page of the KAOS newsletter Vol.3 No.2 the fee for those residing in Australia is $15, New Zealand is $A20 and America and Europe $A25.

From feedback we received, some members at the January meeting misunderstood Ray Gardiner's explanation of what was happening with the Rabble 65 computer. To bring everyone up to date on what is happening, we asked Ray and Bill to write an article explaining the new design. You will find this very interesting article on page 6 of this newsletter.

Ed Richardson passed on to us an excerpt from a letter he received from Eric Lindsay who has just returned from America. Eric stated that OSI MACOM were selling off Superboards at bargain prices, $150 each or 3 for $300, (Eric now owns two Superboards). The latest news is that MACOM has sold OSI to KENDATA and it looks like KENDATA are interested in getting back into the personal computer market, so anything could happen. Saga continued next month....

The next meeting will be held at 2pm on Sunday 27th February 1983 at the Essendon Primary School on the corner of Raleigh and Nicholson Streets, Essendon. The school will be open from 1pm to 5pm.

The closing date for articles for the March newsletter is the 13th March.

INDEX

THE BEGINNING MACHINE LANGUAGE PROGRAMMER....Part 8

by David Dodds

If you were expecting to see lots of code for the promised graphic composer in this issue -think again. This month we have more important things to do like set down a system specification and do the design work. Because of space limitations 'ARTIST', which is what I plan to call the program, will be kept fairly simple. Also because it is good programming practise (and easier to present in a series of articles), ARTIST will be highly structured.

The following is the basic requirement:

- .ability to control the placement of graphic characters anywhere on the screen.
- .Program to support both C1/C2 and C4 screen and keyboard formats.
- .Ability to save and reload screens of composed images to/from tape.
- .Available graphic characters to be displayed on screen.
- .Ability to erase screen.

To achieve this we need some sort of 'paint brush' which could take the form of a mobile cursor. A blinking cursor that can be seen in the middle of a group of characters would be best.

Displaying all of the 256 available graphic characters on a 32 or 24 character screen would seriously limit the size of our artistic efforts. If we were to display say one line and provide controls to allow all characters to be shifted across the screen (like viewing them through a window) we could maximise the size of our 'canvas'.

As the standard input routine on OSI machines sits around waiting for a key to be pressed this would interfere with the blinking cursor requirement. That leaves only the technique of polling the keyboard from the program as the means of obtaining commands. The technique for doing this is well known so we won't dwell on that here.

The way I decided to tackle this program was to start from the top level (the design level) and expand down into the program in a systematic way. In order to start in on the design of ARTIST we could regard the program as comprising three modules:

Initialise artist
Make pictures
Terminate artist

By taking each module in turn and developing these into successively smaller tasks, an overall design will emerge which can be coded directly. For example initialise artist becomes:

.set system dependant parameters	SYSDAT
.initialise other imformation	SET.UP
.clear screen	CLRS
.display window	WINDOW

If we wished we could even code this directly as a series of subroutine calls. Coding the last 2 as subroutines makes sense as both will be needed in other parts of the program. Having SYSDAT and SET.UP as subroutines allows them to be coded last once all the program requirements are known.

The process terminate artist is really only a means of directing ARTIST someplace else and the monitor is as good a place to go as any so the task can be coded directly to JMP $FE00 (GOTO MONITOR).

Obviously make pictures will require a greater amount of expansion. The first important feature of make pictures is that it is a loop - repeat...make pictures...until command equals 'exit'.

Within the module the elements which need to be considered are:

.Get a key
.Move cursor
.Update window
.Erase screen
.Paint a character
.Save picture
.Load picture
.Blink the cursor

All of these tasks except the blinking cursor and get a key are done under keyboard command. Let us assume that get a key will only return once a legal command has been issued. This will mean that the cursor will need to be flashed during the execution of the get a key routine. Also it would be a good idea if the cursor were to flash at least once in between key commands so that we can see where it is during repeated key operations. Because of timing problems two seperate routines will be required for blinking the cursor.

Inspection of the activities to be performed under keyboard control shows the need for 11 keys (cursor 4, window 2, and one each for save, load, erase screen, paint and terminate.
Let's allocate the keys as follows:

U, D, .(>) and ,(<) for cursor controls
S and L save and load
RUBOUT to erase screen
ESC to terminate
F and B (forward and backward) for window controls
P to paint.

Putting together all that we know about make pictures reveals:

```
Repeat
     GETKEY;
          IF KEY ='U' THEN CURSOR-UP;
          IF KEY ='D' THEN CURSOR-DOWN;
          IF KEY ='.' THEN CURSOR-RIGHT;
          IF KEY =',' THEN CURSOR-LEFT;
          IF KEY ='P' THEN PAINT-CHARACTER;
          IF KEY ='S' THEN SAVE;
          IF KEY ='L' THEN LOAD;
          IF KEY ='F' THEN WINDOW-RIGHT;
          IF KEY ='B' THEN WINDOW-LEFT;
          IF KEY ='RUBOUT' THEN CLRS;
     FLASH-CURSOR-ONCE;
          until Key ='ESC'
END MAKE-PICTURES.
```

At this stage then we have completed the top level of our design. Each of these statements can be readily converted to assembly language using the instruction COMPARE (which will be covered next month) coupled with branches around the various subroutine calls.

NEWS

Mr Greg Dubois, has 30 charter memberships in The Australian Beginning for sale at $25 each. There is an extra sting associated with the TAB in that you need to pay $35 for a program to use with the Superboard which enables you to load programs. There are also STD charges to pay if you don't live in Victoria.

Ed Richardson

Superboard

February 1983

Newsletter of the Ohio Superboard User Group, 146 York Street, Nundah. 4012.

FIRMWARE REVIEW - BASIC 5 by Ed Richardson.

In KAOS 3/3, a mention of Basic 5 was made. I had written a small program with a few of the functions in a Past OSUG Newsletter with the thought of a later review. Basic 5 is a program originating in the U.K. which gives your UK101 or Superboard some high speed graphics functions.

Basic 5 modifies the parser routine to add between 15 and 18 new words to the OSI Basic language. The technique is nothing new, and several programs using it have appeared in Micro magazine and elsewhere. The new words can be incorporated in program lines or used as immediate commands. The key letter that distinguishes Basic 5 commands from others is &.

Here is a brief summary of the new words:-

&VLIN and &HLIN plots vertical or horizontal lines using any specified character in the Superboard's character set.

&SET - puts any character at any screen location.

&TEST - gets the number of the character at any specified screen location and makes it a variable.

&SCR - provides an instant screen clear or fill capability.

&BLK - allows instant generation of a block of specified characters.

&GET - a non-halting get character from keyboard routine.

&INAT - inputs a string starting from a specified screen location.

&PUTAT - a printat function, non-scrolling, anywhere on the screen.

&PRNTUSNG - a print using function allowing user specified output formats for printed strings and variables.

&GO and &GO$ - a new way to call a M/Ç routine with decimal or hex address.

&GT and &GS - goto or gosub, only a variable or expression can be used to get the line number.

&RD - will find any numbered piece of data out of a data statement and make it a specified variable.

&WI, &CWI, and &CWI$ - only available in the Cegmon Monitor version, to allow fast manipulation of the print window.

Virtually all of the above functions work under direct number or variable control, or on evaluation of an expression. The graphic functions all operate instantly - ZAP - just like a M/C screen clear.

Basic 5 is available as an Eprom, located at $9000 or on disk (24 or 32k). A tape version is not available. Synmon or Cegmon monitor must be specified when ordering, as well as your system, UK101 or Superboard, and screen format.

If you order it as an Eprom, and don't have a Tasker Bus or Rabble board, then the easiest way to install it is by using Bert Patterson's excellent little Eprom Extender board. It was reviewed in KAOS 2/11 SUPERBOARD newsletter.

Basic 5 can be ordered from Premier Publications, 208 Croydon Road, Anerley, London SE20 7YX. The Eprom costs 20 Pounds and the disk is 18 pounds. This price includes VAT which isn't payable on exports, but the difference will help cover the Air Mail costs. Beware the vicious sales tax sting should Australian Customs intercept the package. A nice letter to Premier could help.

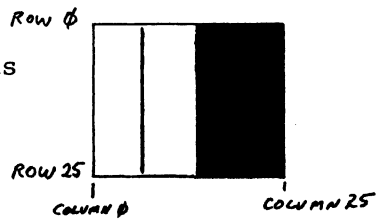
Each Eprom and disk has a 2 byte code somewhere in the 2k to identify the owner and this can obviously mark 65535 buyers. Premier threaten to sue you if copies including your code are detected , thus protecting their interests.

— SUPERBOARD —

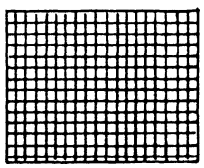
Probably the easiest way to illustrate the workings of Basic 5 commands is to write a program using some of them.

```
10 A=10:&SCR32 :REM FILL SCREEN WITH CHR$(32)- CLEAR SCREEN
20 &BLK0,12,8,25,161:REM ROW,COLUMN,LENGTH,HEIGHT,CHARACTER
30 &VLIN0,5,25,140:REM ROW,COLUMN,HEIGHT,CHARACTER
40 &GO$FD00:&GT A*6:REM WAIT FOR KEYPRESS THEN GOTO 60
60 &SCR209:END:REM FILL SCREEN WITH CHR$(209)
```

Result of lines 20 and 30



Result of line 60



PRINT YOUR OWN CALENDAR

Way back in July 1980, Alan Cashin published a program to print on the screen any calendar month, accurate from 1900 to 2099, and with a neat border, in our OSUG Newsletter. I have modified it slightly to work with a printer to produce a full year. Now you can have that 1984 calendar anytime you want it. Also, as the program prints on the left of the paper, there is loads of room to write in notes and appointments on the remainder of the sheet. Note how the program prints the month name centrally over each calendar month.

```
100 PRINT:PRINTSPC(5)"CALENDAR  YEAR":PRINTSPC(5)"by Alan Cashin"
110 PRINT:PRINT:INPUT"YEAR (1980)";N:PRINT
115 POKE 517,1:FOR MO=1 TO 12:M=MO:RESTORE
120 FOR S=1 TO M:READ A$:NEXT:PRINTSPC((19-LEN(A$))/2)A$;N
130 PRINT"  SU MO TU WE TH FR SA"
140 GOSUB 200:Y=X:M=M+1:IF M>12 THEN M=1:N=N+1
150 GOSUB 200:S=X-Y:X=Y-INT(Y/7)*7:P=X*3+2:Z=8
160 FOR D=1 TO S:A$=STR$(D):IF D>9 THEN A$=RIGHT$(A$,2)
170 IF X>6 THEN X=0:P=2:Z=Z+2:PRINT
180 PRINTSPC(P)A$;:P=1:X=X+1:NEXT:PRINT:PRINT
190 NEXT MO: POKE 517,0:RUN
200 V=M+1:W=N-1900:IF V<4 THEN V=V+12:W=W-1
210 X=INT(V*153/5)+INT(W*365.25)+1:RETURN
220 DATA JANUARY,FEBRUARY,MARCH,APRIL,MAY,JUNE,JULY
230 DATA AUGUST,SEPTEMBER,OCTOBER,NOVEMBER,DECEMBER
```

SOFTWARE REVIEW - POKER

Poker, by Paul Joviak, is on the OSI label. It is in the Library.
You play against the computer, the game is three card draw poker, and if you like winning, then this is the game for you!

At first,I felt the game was a little unfair because you have no idea of the number of cards the computer has drawn. However I found the computer was a regular bluffer and it plays a terrible game. I had no difficulty in beating it in every session that I played. If I had a terrible hand, I simply bet 0 on it. If the computer also had nothing, it folded, leaving me with the ante.

It is usually body language that gives away your hand, good or otherwise, and you need to be a Stonewall Jackson to bluff in a real game. You would think that a computer would make a perfect bluffer, but the program wasn't really up to it, and the hand could nearly always be guessed by the betting.

The program is almost 8k long, and I'm sure some member with an interest in the game would be able to improve on it. The card graphics are nicely done.

Ed Richardson.

THE RABBLE 65

Much has been said and opinions offered towards the Rabble 65 single board computer. We at Rabble Ozi Computers have welcomed the input provided by KAOS members and used this information to provide the final version of the single board computer "RABBLE 65". You may recall that the expansion board which has proved to be so popular had two prototype versions before it was released with all the options you, the members, requested. Since the last reference of the Rabble 65 in KAOS V3, No.3 we have decided to upgrade the basic single board system to provide a fully operational disk system without having to add any further boards.

The total available RAM on the board is 52K and ROM totals 8K. The rest of the 64K is used up with 2K I/O and 2K not allocated. It should be noted that the serial I/O's at $F000 and $FC00 are decoded and used on our system. This provides compatability with C1, C4 and C8 computers.

MEMORY MAP

$0000 - $BFFF	user and system RAM
$C000 - $C7FF	I/O inc. disk, keyboard, 16 pin I/O
$C800 - $CFFF	system and user RAM (BASIC protected)
$D000 - $D7FF	video RAM
$D800 - $DFFF	not allocated
$E000 - $EFFF	user ROM
$F000 - $FFFF	system monitor (RA 65)

DISK I/O: This is located at $C000 (PIA) and $C010 (ACIA), and is able to access up to four drives in single or double sided format. 8" or 5.25" drives may be used. The motor control and data separator are provided for. The motor control function saves an enormous amount of media and head wear on the DC motor drives. Rabble 65 will boot up on the standard OSI software packages, OS65D V3.2, OS65D V3.3, OS65U etc.

RS232: This I/O is provided with true RS232 levels (+12V and -9V), a selectable baud rate generator is provided to suit your printer speed, in the range of 75 baud to 9600 baud. The port is located at both $F000 and $FC00.

16 PIN I/O BUS: The 16 pin I/O bus provides a memory block of 16 locations to drive peripheral devices. The locations used are $C700 - $C70F, and this can access 8 ACIA's, 4 PIA's or a VIA. Other devices can be driven on this bus, some of the currently available hardware to suit this port is a stand alone Hi-Res board , EPROM programmer, Vortrax and a high speed D/A and A/D board. Other hardware options will be available in time, these will include programmable sound generators, voice recognition, VIA, PIA, ACIA and a colour high resolution board based on the Thompson CSF chip, which supports 48K of dynamic RAM.

VIDEO DISPLAY: Rabble 65 uses a 6845 CRTC to produce the video display. The display format is 64 characters by 32 rows with an Ohio compatible character generator, providing 256 characters. A user optional display of 80 X 24 is available for those wishing to use the Rabble 65 as a serial terminal.

The video output is a composite video signal suitable for coupling directly to a video monitor via a RCA connector. It is recommended that a monitor be used rather than a converted TV set, so as to provide the resolution that the computer is capable of providing. For the technically minded readers the dot generator is around 16MHz.

KEYBOARD: A 62 key keyboard will be used on the system and the full upper and lower character set will be supported. The keyboard is of American origin and has reliable contacts and a good feel to it, we tried dozens of boards before selecting this one. The printed circuit board is being redesigned to fit this keyboard. Extra characters that appear on this keyboard make it UCSD Pascal compatible.

CPU: The Rochwell CPU chip, the 65C02, will be used in the computer when sufficient supplies become available. The system clock will run at 1MHz in systems with 1MHz I/O devices. Users may upgrade their I/O devices and also the clock speed to suit.

The system software will start off to be 6502 compatible, and will later develop using the more powerful and flexible commands of the 65C02.

POWER SUPPLY: This will generally be determined by the peripheral devices that the user intends to operate. The following requirements are given to provide a supply which will provide for all anticipated expansion to the system.

+5V 5A +12V 1A +24V 2A (8" Drives) -9V 100mA

PACKAGING OPTIONS: We intend to provide a Keyboard enclosure together with a variety of disk enclosures which will house the CPU board. Refer to the diagrams to see the single board layout and package options.

SOFTWARE OPTIONS: The board will be supplied with a 2732 containing the RA 65 monitor programme. The monitor consists of keyboard and video drivers, disk boot, interrupt handlers and an extended monitor which will include trace facilities. A full source listing will be provided as detailed in the section covering support documentation.

An alternative monitor will be offered for those wishing to operate their Rabble 65 as a serial smart terminal. This monitor will only provide ascreen format of 80 X 24.

DESIGN PHILOSOPHY: It is our aim to provide a low cost disk based computer which is Ohio disk compatible. The system should be suitable designed to replace the C1P and the C4P and be comparable with the C8 computers. We believe we have achieved this in that the system will run all the currently available Ohio compatible software available to us, for a much lower cost than the C1P MF, and includes more features.

Some of the software we have tested, includes:- OS65D V3.2, OS65D V3.3, nical Products FORTH, Softech UCSD Pascal, WP6502 and many others. It is our aim to provide the user with any software support for the disk based computer where the system monitor is involved.

SUPPORT DOCUMENTATION: Three manuals will be available to support the Rabble 65 computer. This will provide easy access to the required information without hunting through non-related text. The manuals are the Technical manual, the System Software manual and the Disk Operating System manual.

TECHNICAL MANUAL: This will provide all hardware circuit diagrams, full and detailed assembly instructions, a description of operation of each section of the computer. All adjustments and timing diagrams will be explained and a comprehensive listing of the 65C02 instruction set will be included.

SOFTWARE MANUAL: A full source listing of the system monitor will be provided with the board and the listing will also be available on disk for those requiring it. All useable subroutines will be listed with a brief description of their function.

DOS MANUAL: This manual will be provided with the DOS disk(s), and will relate to the operating system the user selects. This is not supplied with the board, but is an optional extra.

VARIATIONS: A number of options are available to the user if he so desires. The board is designed as a disk based system, but there is no reason why it cannot be used as a cassette based system if required. After all we have provided a cassette port to Kansas City standards. As we have stated earlier, we will support only software for the disk based system, however much software is currently available to run with BASIC in ROM.

A BASIC interpreter similar to the one used in the Ohio systems, but with added features will be available. The added features will include a clear screen, CLS; trace facilities, TRON and TROF; auto line number, AUTO; and a renumberer.

We are currently pursuing a version of FORTH which will run in ROM. This system will operate similar to the disk based FORTH, but will use RAM as the equivalent of the disk storage facilities.

Other options of course will include the use of PICODOS for those with disk systems and wishing to operate with BASIC in ROM.

Screen format changes can be selected by the user if desired. The standard is 64 X 32, which is C4 and C8 compatible, including the character generator.

A dedicated serial terminal can also be provided. This option is supported with all the necessary serial terminal software in ROM with a selectable baud rate generator covering the rates of 75 - 9600bd. A routine to drive a serial printer port is also included. The screen format for the serial terminal is 80 X 24.

PRICE AND AVAILABILITY: Prototype boards will be available during March. Development boards which we have been using have operated successfully since last November. However, due to the large scale upgrading of the boards facilities only a small number of boards will be produced on the first run (we call these prototype boards). We forsee no problems with the design and expect to have commercial quantities available in April. We suggest you keep an eye on KAOS for further developments.

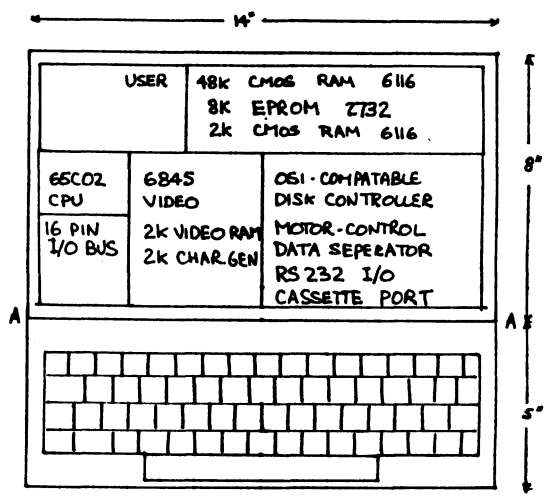
Costing of the system is yet to be finalised, we are still haggling with suppliers for the best price. However, a minimum system can be assembled for under $300, not including the power supply and video monitor.

The bare kit will include the hardware and software manuals, monitor ROM, character generator, keyboard and the computer board. The estimated cost is $140.

RABBLE-OZI-COMPUTERS FEB 83

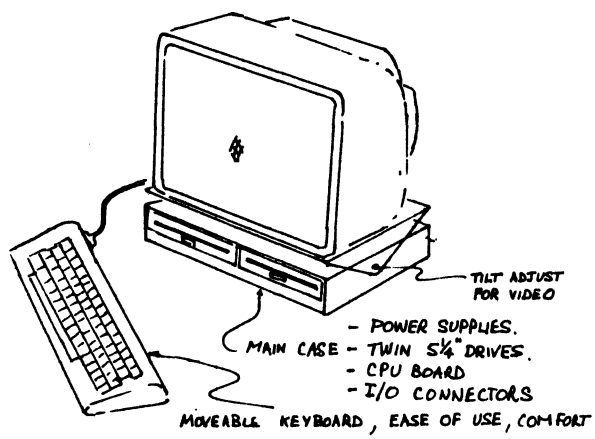
RABBLE 65 SINGLE-BOARD OSI-SOFTWARE COMPATABLE COMPUTER.

BOARD LAYOUT (DIMENSIONS)

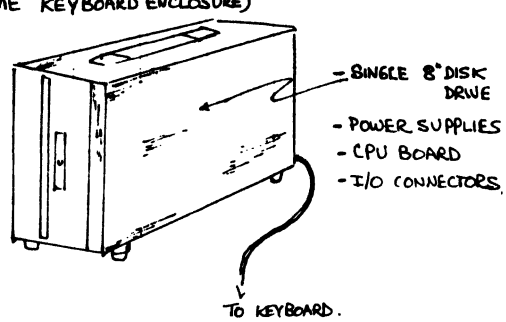


1. THE KEYBOARD MAY BE DETACHED (A-A) AND CONNECTED TO CPU BOARD USING DB-25 CONNECTORS AND 16 WAY CABLE.
2. CONNECTION TO THE FLOPPY DISK IS VIA A 50-WAY CONNECTOR, COMPATABLE WITH 8" DISK DRIVES (MAY BE CONFIGURED FOR 5¼" DRIVES IF REQD)

PACKAGE OPTIONS FOR THE RABBLE 65.



ALTERNATE MAIN CASE
(USE SAME KEYBOARD ENCLOSURE)



THE MEETING WAS KAOS

by Micheal Lemire

King Corky has been usurped from his position in the Ministry of Meeting Reporting; this month I'm running the show.

The January meeting was attended by around 60 people, and there were several developments. The much awaited Rabble 65 will not be available for around 2 months; the prototype is running happily but Ray and Bill have been modifying the computer to have 48K of RAM on board, and have replaced the original data seperator with the more readily available 8602 chip; the circuitry is being changed to match. This enterprising pair have also announced that they are building a stand-alone ie. intelligent, graphics board with 48k of video RAM. That makes 512 X 256 pixels in 8 colours, it should be a great machine.

Jeff Kerry demonstrated his SC-01 speech synthesiser and a talking calculator program he has written. The system could be understood the first time it spoke, opening the meeting and complaining of a pain in the diodes down it's left side. Jeff has added the board to his turtle to aid in his teaching.

OSI (the company that made the Superboard etc, in case the lack of support fooled you) has changed hands again and are reported to be looking at new computer designs, someone suggested that Ray and Bill should show them the Rabble 65.

Elector magazine has designed a 6502 computer and have based their disk interface on the OSI disk controller because it is the best (simplest?) they could find.

David Anear announced that the 40 pin buss version of the 16 pin I/O board and an EPROM burner to plug into it, are now available. The EPROM burner can read and program any thing up to the 2764. It won't handle Motorola's rather individual style of EPROM (but who uses them anyway?). There is also a Tasker buss version of the I/O board.

The 65C02 series CPU's covered in the last newsletter are now available in the U.S. No information yet on when they will be available this side of the Pacific.

The February meeting is to include a garage sale, so you can bring along, and get rid of, all the junk you bought at the last one.

SPEEDING UP RANDOM ACCESS FILE 'GET' FUNCTION

by Nigel Bisset

The following is a short patch which may be added to any progam which uses Random Files under 65D Version's 3.0 and 3.2.

When you use the DISK GET command the DOS calculates which track of your file the requested record is located on then loads that track into the disk buffer,it then sets the memory I/O pointers to the start of the record requested ready for your INPUT#6, A$ or whatever. The problem is that if the next record you ask for is on the same track the DOS reloads that track into memory again. This can be very time consuming and causes unnecessary wear of the disk drive.

The following program checks if the required record is already in memory and if it is, adjusts the I/O pointers and returns, no redundant disk load is executed. If the record is not in the buffer then it checks the disk buffer Dirty Flag at 9005 Dec. to see if you have changed aything. If you have it writes the buffer to disk with a PUT command then loads the next track required.

USING THE PATCH.
Open the file as usual - DISK OPEN,6,Filename.
Set record size with two pokes if not using default record size of 128 bytes
Set the variable 'RN' to the number of the record you wish to access.
GOSUB 10000 - Transfer control to the patch program.
When finished close the file as normal - DISK CLOSE,6

```
10000 REM  PATCH TO SPEED UP DISK GET FUNCTION
10010 DEF FNS(X)=10*INT(X/16)+X-16*INT(X/16)
10020 TR=INT(RN/PEEK(12042))
10030 IF TR+(FNS(PEEK(9002)))=FNS(PEEK(9004))THEN10060
10040 IF PEEK(9005) THEN DISK PUT
10050 DISK GET,RN:RETURN
10060 RA=(RN-TR*PEEK(12042))*(2↑PEEK(12076))+PEEK(8998)+PEEK(8999)*256
10070 AH=INT(RA/256):AL=RA-AH*256
10080 POKE 9132,AL:POKE 9133,AH:POKE 9155,AL:POKE 9156,AH:RETURN
```

This program is based on a patch program published by OSI in the Small System Journal November 1980. Unfortunately their version only worked correctly if the track number was less than 9, on tracks higher than this it failed.

FORTH DISK SIMULATION

by Brian Campbell

Anyone who has tried using FORTH without a disk will have found it extremely frustrating. Entering word definitions must be done directly from the keyboard and therefore they can not be listed or corrected. Saving these routines is a messy process involving the manual fiddling of pointers and usually requires saving the FORTH language on tape as well.

This article deals with the use of RAM to simulate a few screens of disk storage. All of the software described here is available from me on cassette for the SYM. Those with other systems will be able to apply the principles to their own computers.

The idea of using RAM to simulate a disk is not new. The 'fig-FORTH Installation Manual' describes how this can be done and gives the new R/W definition. Unfortunately, it does not work very well when using screen 0. After months of checking and rechecking code believing I had made an error in implementing the new routine, I discovered an oversight which meant that the R/W routine presented could never work properly in screen 0. For the benefit of anyone else who may be losing sleep over the same problem, I will explain the reasons for the fault.

The most obvious fault occurs when LISTing screen 0 after typing EMPTY-BUFFERS. The first two lines of the screen will not be printed. This is because the word EMPTY-BUFFERS fills the disk buffer area with zeros. This suggests that all of the buffers contain blosk 0 which consists of nulls. When '0 LIST' is typed, FORTH will first look for block 0, which contains the first two lines of the screen, using the word BLOCK. This will only load a block from disk if it does not already exist in the buffers. Although block 0 does not exist in the buffers, EMPTY-BUFFERS indicates otherwise. This means that the real block 0 will not be loaded and the 128 null characters will be used instead.

Even if the above mentioned problem were to be solved, another bug would appear. Assuming that valid source had been entered into screen 0, the command '0 LOAD' would not function correctly. The word LOAD calls up INTERPRET which in turn calls -FIND and finally gets to WORD. This word copies the next word from the input stream to PAD. The input stream is determined from the block number on the disk that the data is coming from and a zero indicates the keyboard buffer. When LOAD tries to interpret anything on the first two lines (block 0), FORTH will fall in a heap.

The solution to this problem is extremely simple; don't use screen 0. For this reason the modified R/W routine will simulate screens 1 to 16 instead of 0 to 15. The new R/W also contains error checking to prevent the use of block numbers corresponding to screens outside this region since the memory locations used could fall anywhere in the 6502's memory map.

Listed below is the new R/W definition:HEX

```
: R/W  >R  (save boolean)
B/SCR -   (offset by one screen)
DUP 0< 6 ?ERROR   (lower end range check)
DUP B/SCR / F > 6 ?ERROR   (upper end range check)
B/BUF * 4000 +
R> IF (read) SWAP ENDIF
B/BUF CMOVE
;
```

These faults do exist with normal disk versions of FORTH, but since the lower screens usually contain system or directory information, screen and block 0 are rarely accessed and are never used for FORTH source statements. The problem of block 0 never arises.

In order to make the use of the RAM disk simple, a number of cassette based words have been added to the dictionary. These are CSAVE, CLOAD and CONTINUED-ON. The functions of the first two are obvious. They save and load screens on cassette. CONTINUED-ON is similar to RAE's .CT directive. This word allows LOADing to continue automatically after loading new screens off tape. The format for these three words is:

```
s  c CSAVE f
s  CLOAD [f]
s  CONTINUED-ON f
```

where

s is the first screen number to be saved or loaded. The word CONTINUED-ON will also begin LOADing on this screen

c is the number of screens to be saved. s+c must be 17 or less. eg. To save the last screen, s=16, c=1, sum=17.

f is the filename. This must be two or more characters in length and must conform to FORTH's restrictions governing words; no spaces, no carriage returns and no nulls. This still leaves the choice of filename fairly free. The filename is optional for CLOAD. If the filename is left off the next file found will loaded, however this should be used only when entered from the keyboard.

All three words are execute only.

A machine code routine called SUPERMON allows FORTH to execute a SUPERMON command. The constants P1, P2, P3, PARNR and LSTCOM are merely aids for using the word SUPERMON. All SUPERMON does is save the registers, load the Zaccumulator with $0D (to keep the monitor happy) and call the monitor subroutine DISPAT. For those unsure of the workings of the monitor, refer to the SYM-POSIUM article in Vol.2 No.7 of the KAOS newsletter.

CSAVE actually generates two files. The first has an ID of 01 and contains the filename and the number of screens saved. The second file (ID=02) contains the actual screen data.

CLOAD searches for a file with an ID of 01 and loads it. If a filename was specified and the two filenames do not match, the routine will search for the next file with an ID 01. If the file found is a correct one, the next file is loadedstarting at the specified screen. Before loading this file, a check is made to ensure that there is enough room for it in memory.

CONTINUED-ON is simple:

```
* CONTINUED-ON
?EXEC   (execute only)
DUP
[COMPILE] CLOAD   (filename supplied as part of CONTINUED-ON)
LOAD   (complile)
;
```

IMMEDIATE
An editor is required to store the text in the simulated disk space. The one available to SYM users is the editor given in the installation manual. It is not a good editor by any means, but it will do until a better one can be written. The editor will be supplied in a CSAVE format and will not be included as a part of the FORTH language itself. This will allow the new editor to be attached with the minimum of effort.

I hope this article has been helpful to those who can not afford a disk drive. If you have any queries or comments I would be happy to hear from you. Good luck!

NOTES FROM THE WEST

At a recent meeting of interested people it was decided to form an Ohio Scientific users group in W.A. Of 22 people known to have Ohio equipment, 13 attended or apologised for their non-attendance. It was agreed that to be a member one should also be a member of KAOS.

Meetings are to be held bi-monthly on the 3rd Sunday of each odd month. The next meeting is on the 20th March at 2.00pm. It is to be held on the top floor of Guild House, 56 Kishorn Rd, Mt Pleasant. You are invited to bring your computer and Peter Hughes has indicated that his printer will be available for those who wish to obtain a listing of their programs.

The meeting will be mainly an informal one so people can share their interests and discuss any problems they are encountering with their system.

Further information can be obtained from Gerry Ligtermuet

USERS of the OSI ASSEMBLER *BEWARE !*

by David Dodds

The assembler does not differentiate between the various fields within an assembly language program, and can't always tell the difference between a label and an illegal opcode.

For this reason some of the new inherant mode instructions in the 65C02, such as PHX, PLY will appear to assemble. However close inspection of the assembly listing will reveal that no code was generated. The assembler has taken the instruction as a label and no error was detected.
While the line 10 PHY will appear to assemble
the line 10 PUSHY PHY yields an error. Because of the label the assembler detected an illegal opcode.

The same problem also appears with directives.
e.g. 10 .FRED appears to assemble. However 10 .$FRED will yield an undocumented error message E# 26 - illegal directive.

RABBLE SOUND GENERATORS

by Paul Dodd

```
 10 C014=           DRA=$C014
 20 C016=           DRB=DRA+2
 30                 ;
 40 0000=           INAC=0
 50                 ;
 60 00F0=           PSG=$F0
 70 00F1=           REG=PSG+1
 80 00F2=           INF=REG+1
 90 00F3=           WRITE=INF+1
100 00F5=           LATCH=WRITE+2
110                 ;
120 3A82           *=$3A82
130                ;
140 3A82 A902    START  LDA #2
150 3A84 85F3           STA WRITE
160 3A86 A908           LDA #8
170 3A88 85F4           STA WRITE+1
180 3A8A A903           LDA #3
190 3A8C 85F5           STA LATCH
200 3A8E A90C           LDA #12
210 3A90 85F6           STA LATCH+1
220 3A92 A900    LDREG  LDA #INAC
230 3A94 8D16C0         STA DRB
240 3A97 A4F0           LDY PSG
250 3A99 B9F500         LDA LATCH,Y
260 3A9C 8D16C0         STA DRB
270 3A9F A5F1           LDA REG
280 3AA1 8D14C0         STA DRA
290 3AA4 A900           LDA #INAC
300 3AA6 8D16C0         STA DRB
310 3AA9 A5F2           LDA INF
320 3AAB 8D14C0         STA DRA
330 3AAE B9F300         LDA WRITE,Y
340 3AB1 8D16C0         STA DRB
350 3AB4 A900           LDA #INAC
360 3AB6 8D16C0         STA DRB
370 3AB9 60             RTS
```

At last, all is revealed (well something is revealed - I'm not sure what). For all those people who wanted to use the Programmable Sound Generators on the Rabble Board here's how to do just that:
Firstly you need a small machine code routine;(I assume you can enter assembly language programs, if you can't - tough!) If you have a cassette machine change line 120 to *=$0222. Now enter lines 10 to 60 and 999 to 1010 of the BASIC program. If you have a cassette machine, change DISK!"... to POKE 11,34:POKE 12,2:X=USR(X). If you have a disk machine, refer to "Embedding Machine Code in BASIC Programs" by Michael Lemaire in this issue of KAOS.

To store data in a register type:
xxx R=reg no : C=data = GOSUB 1000
An example is given in lines 100 to 160 of the BASIC program. This should give a gun-shot effect.
NOTE: If you don't get any sounds when you are programming the PSG's check to see if you set the amplitude registers before swearing at the AY3-8910 chips!

```
10 PIA=49172
20 POKEPIA+1,0:POKEPIA+3,0
30 POKEPIA,225:POKEPIA+2,255
40 POKEPIA+1,4:POKEPIA+3,4
50 PSG=12153+240:REG=PSG+1:INF=REG+1
60 POKEPSG,0
100 R=7:C=199:GOSUB1000
110 R=6:C=17:GOSUB1000
120 R=8:C=16:GOSUB1000
130 R=9:C=16:GOSUB1000
140 R=10:C=16:GOSUB1000
150 R=12:C=9:GOSUB1000
160 R=13:C=0:GOSUB1000
999 END
1000 POKEREG,R:POKEINF,C: DISK!"GO 3A82"
1010 RETURN
```

EMBEDDING MACHINE CODE IN BASIC PROGRAMS

by Michael Lemaire

Often it is useful to call a machine code routine from a BASIC program to carry out some function that needs to be done quickly. It is possible to modify pointers in BASIC to provide area for the code which is saved as part of the program.

In OS65D V3.3, all source files are loaded to, and saved from, $3A7D. The first two bytes point to the null at the start of the BASIC program, which usually starts at $3A7F. This pointer is copied down to locations 120 and 121 (decimal). If the pointer at 120-121 is increased, a BASIC program subsequently entered will start at that higher address, and a hole is left from $3A7F up to the start of the program; a convenient place to put machine code. Since the system always PUts files from and GEts files to $3A7D, the machine code goes anywhere that the BASIC program goes.

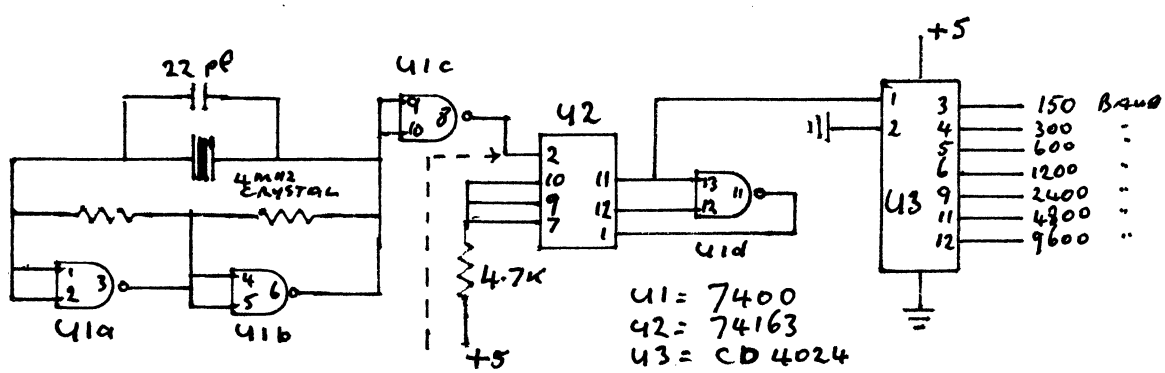
SETUP PROCEDURE:
(1) Assemble the machine code routine to $3A80 or thereabouts.
(2) Save the assembled code to a file using the minimum number of pages; eg. a small program which fits in a single page can be saved by 'SA tt,1=3A80/1' (where tt is a track number).
(3) Boot BASIC and then alter the 120-121 pointer by RUNning 'CHANGE' or manually poking (be careful!).
eg. Usually in V3.3, PEEK(120) is 127 and PEEK(121) is 58. PEEK(121) is the high byte. To set up a hole of, say, 258 bytes (1 page plus 3 bytes for luck), use the statement:
POKE120,130:POKE121,59:NEW
The 'NEW' is necessary for BASIC to put a null in the start of the program area; otherwise you'll get a terminal mess.
(4) Load in the machine code with 'CA 3A80=tt,1'. The code is now embedded in the BASIC source file.
(5) Enter the BASIC program by (i) Keyboard; (ii) Using an ASCII file, or (iii) using indirect files. You cannot just LOad the program above the machine code, as this will reset the source file pointers from the information in the file header.
(6) Save the program and the machine code; The End.

PASSING PARAMETERS:
It is useful to be able to pass parameters to the machine code routine from BASIC; for instance in a memory-shift routine (for reverse scroll, maybe?) it might be convenient to pass the source address, destination address, and number of bytes to be shifted to the routine. This can be done via the Z-page. However, you don't poke to the Z-page itself, but to the page 0/1 swap buffer at 2F79 (12153 decimal), as when a DISK!"GO " statement is executed, these pages are swapped back. This makes both pages fully available to the programmer while preserving all the system stuff. For example,
To place a value in location $F0 (240 decimal), poke it to 12153 + 240.
NOTE:
In OS65D V3.2, the source file starts at $327F, instead of the V3.3 address of $3A7F.

BAUD RATE GENERATOR
by David Tasker

PLEASE NOTE: David Taskers phone no. is

A recent KAOS article presented a single chip Baud rate generator using the Motorola MC14411 and a 1.8432 MHz crystal. Presented here is a less expensive alternative. The MC14411 and crystal pair could cost up to $25.00, that is if you can find a supplier. This little circuit, whilst admittedly using more chips and wiring will cost around $8.00, which includes $4.00 for a crystal. You can however derive the time-base from your C1, C4 or any computer with a 4MHz point.



Alternate 4MHz input from C1P or other divider chain

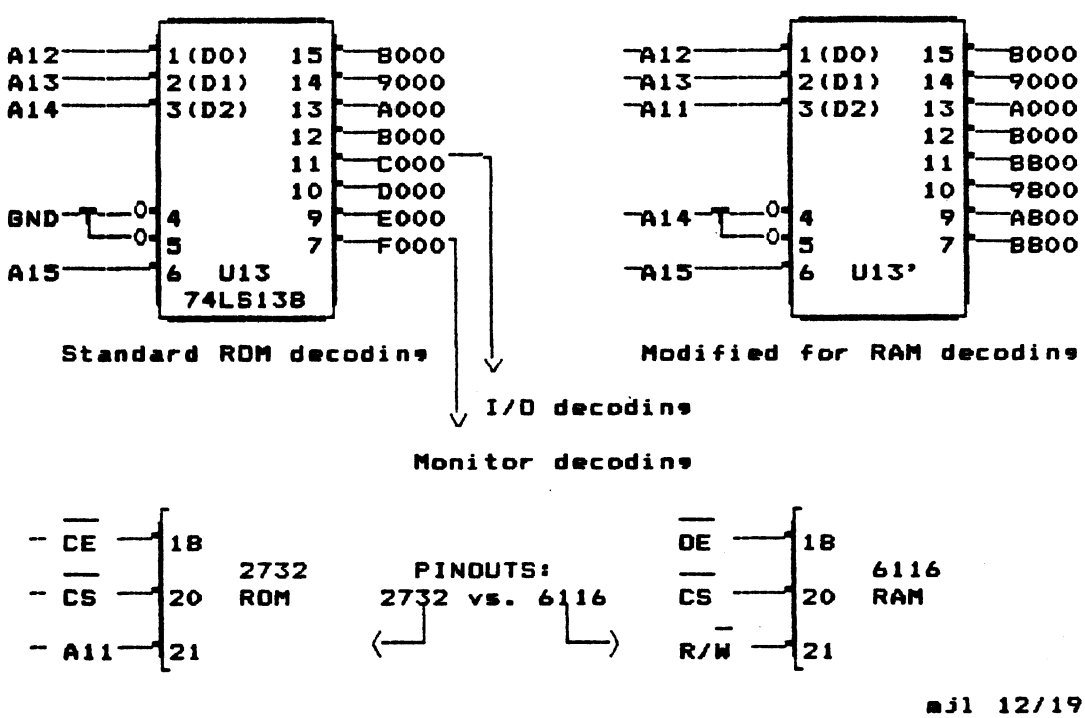
THE INAUGURAL RABBLE BOARD MODS
(or how to build a C8P without really trying)
by Michael Lemaire

After building the Rabble Expansion board and running it for a few months with a Superboard and Tasan Video board, I one day looked at the board and thought, why not do something with the unused 32K worth of ROM sockets? I decided to modify the decoding to run another 16K of CMOS RAM. This turned out to be very easy; all you need to do is change three inputs to the U13 decoder (74LS138). Cut the tracks to pins 3, 4, and 5, then link A11 to pin 3 and link A14 to pins 4 and 5. Also cut the track from U13 pin 11 to U12 pin 4. This line is used for I/O decoding; if left, the disk will migrate to 8800, making little things like booting up impossible. This changes the decoding as in figure 1. The tracks to the sockets had to be changed to supply R/W and different Chip Selects. Remove the BASIC ROMs to prevent clashes with the RAM from A000 to BFFF. At this point a C8P monitor will be needed as C1 and C4 monitors call a print-message routine in the BASIC 4 ROM to output the D/C/W/M message. (Besides, C and W don't have any purpose once the BASIC ROMs are removed.)

Of course I now had no decoding for the I/O section (a slight disadvantage), so I piggy-backed another LS138 with the original inputs, and linked pin 11 (Cxxx) to U12 pin 4.

At this point the only things being used on the Superboard were the CPU, monitor and keyboard. The CPU was the first to go. I wire-wrapped the board with two forty pin sockets alowing me to hang the CPU off the forty pin bus. Two extra lines are needed to the CPU which are not available on the bus -- a +5V line and a phi 0 clock. I used two of the ground pins, 8 and 9 for these lines; I cut off the ground connections and supplied the wanted lines from the Rabble board. Phi 0 is available from U58 pin 4 (1MHz) or pin 3 (2MHz). This means that one of the sockets on the Rabble board can now only be used with the CPU board.

RABBLE BOARD DECODING FOR 48K RAM



Next the monitor moved to the Rabble board. I stuck a 24 pin socket on the prototype area and linked all the address and data lines from the adjacent line of RAMs, and supplied an Fxxx Chip Select from the piggy-backed LS138 (pin 7).

Finally the only thing on the Superboard being used was the keyboard. I put two chips, 74LS00 and 74LS138 on the proto area to decode DFxx, RKbd and WKbd signals. Two 74LS75 quad latches, two 74LS125 quad tristate buffers, eight 1N914 diodes and some pulldown resistors completed the keyboard which uses the same circuitry as the Superboard, modified for C4P.
- And good-bye Superboard.

FOR SALE

Due to price reductions for ICs and memory chips, COMP-SOFT has slashed the prices on OHIO compatible boards to KAOS members.

RABBLE OZI EXPANSION BOARD

BOARD ONLY	$85	32K RAM	$100
ASST. PARTS	$45	16K RAM	$50
I/O and SOUND	$40	8K RAM	$25
	ASSEMBLED & TESTED $70		

TASAN VIDEO BOARD

BOARD ONLY	$40	FULL KIT	$115
PARTIAL KIT	$95	DABUG UPGRADE	$10

CABLES FOR BOTH BOARDS $15 ea. P & P (overnight) $5

COMP-SOFT MICROCOMPUTER SERVICES, 235 SWAN ST, RICHMOND 3121. PH.(03)428 5269

8" double sided disk drives - QUME DT8, 2 off, new $1100.00 the pair.

Tasker buss hardware: 2 8K RAM cards $50.00 each, I/O card $35.00, EPROM card $25.00, Mother board fully socketed $50.00, Metal case complete with card runners $30.00. Discount for the lot.

OSI microcomputer. Runs C8 software. Consists of: Superboard II, Tasan video board, Rabble Ozi expansion board, two 8" double sided disk drives. Facilities include: 48K RAM, PIA, VIA, PSG, colour RAM (inverse video), 10 amp power supply, modified B&W TV, Multiple speed cassette port, parallel and serial printer port. New cost in excess of $2700.00. As a complete working machine will sell for $1790.00. Call Peter on:

Superboard, Series II, Dabug III, 1-2 MHz, 300-2400 baud in-built cassette recorder, RTC, 16K RAM, 10A power supply with fan, all built in a fibreglass case, with over 100 programs, & 14" B/W monitor. $599.00 Gavin Eakins

---

OSI computer with 600, 610 and Tasan Video boards (with inverted keyboard and C4P Dabug, to emulate C4P), 32K static RAM, all in metal case. Also 5.25" MPI floppy, power supply. VDU, Pascelf, Comp-Dos 1.2 and several games. Full documentation $1600 ono. Phone Warwick

---

One 8K Tasker RAM board, with header card still attached, so suitable for connection to 40 pin socket without motherboard. No bugs. OK at 2MHz. $40.00 Bernie Wills,

---

3 ASSEMBLED and TESTED 8K TASKER RAM BOARDS $110